



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/028,739	12/21/2001	Sander Bogdan	MS1-870US	9907
22801	7590	12/27/2005	EXAMINER	
LEE & HAYES PLLC 421 W RIVERSIDE AVENUE SUITE 500 SPOKANE, WA 99201			RAMPURIA, SATISH	
			ART UNIT	PAPER NUMBER
			2191	

DATE MAILED: 12/27/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 10/028,739	Applicant(s) BOGDAN ET AL.	
	Examiner Satish S. Rampuria	Art Unit 2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 05 October 2005.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-41 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-41 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. _____.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

Response to Amendment

1. This action is in response to the Amendment received on Oct. 5, 2005.
2. The rejection under 35 U.S.C. §101 to claims 25-33 is withdrawn in view of Applicant's amendment.
3. Claims amended by the Applicant: 25.
4. Claims pending in the application: 1-41.

Drawings

5. The drawings (Fig. 8) were received on Oct. 5, 2005. The drawing is acceptable by the examiner.

Response to Arguments

6. Applicant's arguments with respect to claims have been considered but they are not persuasive.

In the remarks, the applicant has argued that:

- (i) Although Keeley refers to a development system (see, col. 2, lines 46-49), there is no discussion or mention in the cited portion of Keeley, or elsewhere in Keeley, of selecting development files that correspond to identified SDK objects. Without any such discussion or mention, Applicant respectfully submits that Keeley cannot disclose or suggest selecting the development files that correspond to the identified SDK objects as recited in claim 1 (Remarks, page 20).

Art Unit: 2191

- (ii) Although Keeley refers to a development system (see, col. 2, lines 46-49), there is no discussion or mention in the cited portion of Keeley, or elsewhere in Keeley, of exporting selected development files to a software development kit. Without any such discussion or mention, Applicant respectfully submits that Keeley cannot disclose or suggest exporting the selected development files to a software development kit (SDK) that supports development of applications for use with the modularized system as recited in claim 1 (Remarks, page 21).
- (iii) With respect to claim 25, Applicant respectfully submits that, similar to the discussion above regarding claim Keeley in view of McInerney does not disclose or suggest the selecting and filtering of claim 25. For at least these reasons, Applicant respectfully submits that claim 25 is allowable over Keeley in view of McInerney (Remarks, page 24).
- (iv) With respect to claim 34, Applicant respectfully submits that, similar to the discussion above regarding claim 1, Keeley in view of McInerney does not disclose or suggest the SDK object generator, the feature identification module, the dependency tracer, and the export module of claim 34. For at least these reasons, Applicant respectfully submits that claim 34 is allowable over Keeley in view of McInerney (Remarks, pages 25-26).

Examiner's response:

- (i) In response to Applicant's argument, Keeley's system provides a development system that allows application programs to be developed in the environment

Art Unit: 2191

of a complex operating system. Where his system select the only modules (development files) those are needed to build the particular operating system (See the Summary). Thus, the system does provide selecting the development files for an identified embedded operating system. Applicant only makes general allegations. Therefore, the rejection is proper and maintained herein.

- (ii) In response to Applicant's argument, Keeley's does not mention exporting selected development files to a software development kit. Keeley does disclose in FIG. 4 developing of application program, where the system uses a scanner to collect (export) a list of operation names used in the application program by comparison to those names to the API (col. 6, lines 17-67 and col. 7-8). Thus, the system does provide exporting feature as claimed in claim 1. Therefore, the rejection is proper and maintained herein.
- (iii) In response to Applicant's argument, claim 25 is computer product claim and is rejected under the same rational as claim 1. For the limitation filtering is similar to exporting the selected development files as disclosed by Keeley not McNerney as rejected in claim 1. Therefore, the rejection is proper and maintained herein.
- (iv) In response to Applicant's argument, the similar discussion as applied to argument (i) above, applies here as well. Applicant only makes general allegations. Therefore, the rejection is proper and maintained herein.

7. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

Art Unit: 2191

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Claim Rejections - 35 USC § 103

8. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

9. Claims 1-3, 4, 6-17, 19-21, 23-27, 29-38, and 40-41 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,138,271 to Keeley (hereinafter called Keeley) in view of US Patent No. 5,325,533 to McInerney et al. (hereinafter called McInerney).

Per claim 1:

Keeley disclose:

- creating a software development kit object (SDK object) for at least some of a plurality of development files in a source operating system that includes

Art Unit: 2191

development files and components (col. 2, lines 62-65 “a software development system including a modular operating system program having a plurality of modules each providing an operating system operation that may be called by an application program”);

- identifying features of the source operating system to be included in a modularized system that is a subset of the source operating system (col. 3, lines 28-30 “identify both operating system modules and portions of those modules needed for the application program”);
- selecting the development files that correspond to the identified SDK objects (col. 3, lines 10-11 “selected modules may be collected and reformed into a smaller operating system”); and
- exporting the selected development files to a software development kit (SDK) that supports development of applications for use with the modularized system (col. 3, lines 4-7 “selective compiler program receiving the OS module list prepares an operating system comprised of only those modules of the modular operating system necessary to perform the application”).

Keeley does not explicitly disclose tracing dependencies in a dependency model correlating to the source operating system that uses the SDK objects to identify SDK objects corresponding to development files that are required to support the identified features.

However, McInerney discloses in an analogous computer system tracing dependencies in a dependency model correlating to the source operating system (col. 3,

Art Unit: 2191

lines 41-42 “system automatically keeps track of editing changes in components” and col. 3, lines 45-46 “Dependency analysis is automatic and is based on relations between components”) that uses the SDK objects to identify SDK objects corresponding to development files (col. 3, line 44 “systems that track only at the file level”) that are required to support the identified features (col. 3, lines 7-9 “A program is modeled as semantic units called components made up of a list of named data items called properties”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of tracing the dependencies files in creating the modeled computer program as taught by McInerney into the method of generating the modular code for operating system as taught by Keeley. The modification would be obvious because of one of ordinary skill in the art would be motivated to trace the dependency of a component in software development to provide efficient programming techniques as suggested by McInerney (col. 2, lines 57-64).

Per claim 2:

The rejection of claim 1 is incorporated, and further, Keeley disclose:

- wherein the modularized system is a modularized system (col. 2, lines 53-54 “modular operating system”) that includes a subset of development files and components of the source operating system (col. 2, lines 62-64 “a software development system including a modular operating system program having a plurality of modules”).

Art Unit: 2191

Per claim 3:

The rejection of claim 1 is incorporated, and further, Keeley does not disclose generating the dependency model using the SDK objects.

However, McNerney discloses in an analogous computer system generating the dependency model using the SDK objects (col. 3, lines 32-34 “compiler generated dependencies to correctly and efficiently sequence the compilation of components during a build process”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of generating the dependency model using the SDK objects as taught by McNerney into the method of generating the modular code for operating system as taught by Keeley. The modification would be obvious because of one of ordinary skill in the art would be motivated to generate dependency model using SDK objects in software development to provide efficient programming techniques as suggested by McNerney (col. 2, lines 57-64).

Per claim 4:

The rejection of claim 1 is incorporated, and further, Keeley disclose:

- creating SDK objects for the development files (col. 3, lines 33-35 “The development system... a builder receiving the modular operating system”); and
- for all the development files (col. 2, lines 62-63 “software development system”), if a first development file depends on a second development file, including a reference in a first SDK object associated with the first development file to a second SDK object associated with the second development file (col. 7, lines 6-10

Art Unit: 2191

“The scanner 50 collects each operating system operation name 40 in a OS module list 53... OS module list 53 includes operating system operations F1 and F2 which are directly referenced in the application source code 20”).

Keeley does not explicitly disclose identifying dependencies between development files.

However, McInerney discloses in an analogous computer system identifying dependencies between development files (col. 3, lines 45-46 “Dependency analysis is automatic and is based on relations between components”). The feature of identifying dependencies between development files would be obvious for the reasons set forth in the rejection of claim 1.

Per claims 6 and 10:

The rejection of claim 1 is incorporated, and further, Keeley does not explicitly disclose tracing references from a first SDK object associated with a feature to at least a second SDK object and, if the second SDK object includes a reference to a third SDK object, tracing the reference to the third SDK object.

However, McInerney discloses in an analogous computer system the tracing dependencies further comprises tracing references from a first SDK object associated with a feature to at least a second SDK object and, if the second SDK object includes a reference to a third SDK object, tracing the reference to the third SDK object (col. 4, lines 12-17 “The build mechanism assumes no preknowledge of dependencies... it “discovers” the dependencies of the components and keeps track of those dependencies... build mechanism will build a program from scratch when there is no preexisting dependency information”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of identifying dependencies between development files as taught by McNerney into the method of generating the modular code for operating system as taught by Keeley. The modification would be obvious because of one of ordinary skill in the art would be motivated to identify dependencies between development files to provide an optimize solution for mapping from source to object cod and vice versa as suggested by McNerney (col. 2, lines 34-56).

Per claims 7 and 11:

The rejection of claim 1 is incorporated, and further, Keeley does not explicitly disclose identifying a data object associated with each identified feature, the data object being associated with a component of the source operating system; and if a data object includes a reference to a first SDK object, tracing the reference to the first SDK object and tracing references, if any, from the first SDK object to a second SDK object.

However, McNerney discloses in an analogous computer system identifying a data object associated with each identified feature (col. 3, lines 29-30 “calculating the dependencies associated with a component”), the data object being associated with a component of the source operating system (col. 1, line 13 “source code editing in a computer program” and col. 1, lines 15-16 “useful in developing complex programs, such as operating system (OS) software”); and if a data object includes a reference to a first SDK object, tracing the reference to the first SDK object and tracing references, if any, from the first SDK object to a second SDK object (col. 4, lines 12-17 “The build

Art Unit: 2191

mechanism assumes no preknowledge of dependencies... it "discovers" the dependencies of the components and keeps track of those dependencies... build mechanism will build a program from scratch when there is no preexisting dependency information").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of identifying dependencies between development files as taught by McNerney into the method of generating the modular code for operating system as taught by Keeley. The modification would be obvious because of one of ordinary skill in the art would be motivated to identify dependencies between development files to provide an optimize solution for mapping from source to object cod and vice versa as suggested by McNerney (col. 2, lines 34-56).

Per claim 8:

The rejection of claim 1 is incorporated, and further, Keeley disclose:

- naming a data object having a type that identifies the data object as being an SDK object (col. 5, lines 6-7 "a standard set of operation names 40 that are linked to particular modules 44 of the operating system");
 - including at least one reference in a first SDK object (col. 5, line 7 "linked to particular modules 44"), the reference pointing to a second SDK object that is required by the first SDK object to function properly (col. 5, lines 6-10 "operation names 40 that are linked to particular modules 44 of the operating system holding corresponding operations, each operation being a routine for performing the desired operation (e.g., a disk read, etc.)");
- and

Art Unit: 2191

- repeating the previous steps for each development file to be exposed in the SDK (col. 5, lines 10-13 “API 28 provides an upwardly compatible interface to the application program regardless of changes and upgrades in the operating system 18”).

Per claim 9:

The rejection of claim 1 is incorporated, and further, Keeley disclose:

- an export list in the first data object that identifies one or more functions that may be exposed by the development file associated with the SDK object (col. 3, lines 19-21 “The scanner program... include in the OS module list subsequent calls from an operating system module (referenced in the OS module list) to other operating system operations”).

Per claims 12 and 23:

The rejection of claim 1 is incorporated, and further, Keeley disclose:

- wherein the exporting further comprises storing the selected development files on one or more computer-readable media (col. 4, lines 21-24 “an application program development, the memory 16 will normally hold both an operating system 18 and an editable version of the application program 20 (termed "source code") as well as other programs,”).

Art Unit: 2191

Claim 13 is the computer program product claim corresponding to method claims 1 and 2 and rejected under the same rational set forth in connection with the rejection of claims 1 and 2 above.

Claim 14 is the computer program product claim corresponding to method claims 1 and 2 and rejected under the same rational set forth in connection with the rejection of claims 1 and 2 above.

Claim 15 is the computer program product claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 16 is the computer program product claim corresponding to method claim 6 and rejected under the same rational set forth in connection with the rejection of claim 6 above.

Claim 17 is the computer program product claim corresponding to method claim 7 and rejected under the same rational set forth in connection with the rejection of claim 7 above.

Per claim 19:

Keeley disclose:

- identifying features in a source operating system to be included in a modularized system (col. 3, lines 28-30 “identify both operating system modules and portions of those modules needed for the application program”);

Art Unit: 2191

- selecting development files to be included in an SDK (col. 3, lines 10-11 “selected modules may be collected and reformed into a smaller operating system”); and
- exporting the selected development files (col. 3, lines 1-4 “A scanner program... produce an OS module list of such application calls”); and wherein the development files are files required to support development of applications to work with the modularized system (col. 3, lines 4-7 “selective compiler program receiving the OS module list prepares an operating system comprised of only those modules of the modular operating system necessary to perform the application”).

Keeley does not explicitly disclose by tracing dependencies in a dependency model beginning with data objects associated with the identified features.

However, McInerney discloses in an analogous computer system by tracing dependencies in a dependency model beginning with data objects associated with the identified features (col. 3, lines 41-42 “system automatically keeps track of editing changes in components” and col. 3, lines 45-46 “Dependency analysis is automatic and is based on relations between components”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of tracing the dependencies files in creating the modeled computer program as taught by McInerney into the method of generating the modular code for operating system as taught by Keeley. The modification would be obvious because of one of ordinary skill in the art would be

Art Unit: 2191

motivated to trace the dependency of a component in software development to provide efficient programming techniques as suggested by McInerney (col. 2, lines 57-64).

Per claims 20 and 24:

The rejection of claim 1 is incorporated, and further, Keeley does not disclose generating the dependency model using the SDK objects.

However, McInerney discloses in an analogous computer system generating the dependency model using the SDK objects (col. 3, lines 32-34 “compiler generated dependencies to correctly and efficiently sequence the compilation of components during a build process”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of generating the dependency model using the SDK objects as taught by McInerney into the method of generating the modular code for operating system as taught by Keeley. The modification would be obvious because of one of ordinary skill in the art would be motivated to generate dependency model using SDK objects in software development to provide efficient programming techniques as suggested by McInerney (col. 2, lines 57-64).

Per claim 21:

The rejection of claim 19 is incorporated, and further, Keeley disclose:

- creating SDK objects for at least some of the development files (col. 3, lines 33-35 “The development system... a builder receiving the modular operating system”);

Art Unit: 2191

- for all development files (col. 2, lines 62-63 “software development system”) that have an SDK object associated with it, if a first development file depends on a second development file, including a reference in a first SDK object associated with the first development file to a second SDK object associated with the second development file(col. 7, lines 6-10 “The scanner 50 collects each operating system operation name 40 in a OS module list 53... OS module list 53 includes operating system operations F1 and F2 which are directly referenced in the application source code 20”).

Keeley does not explicitly disclose identifying dependencies between development files; and wherein the identifying and creating generates the dependency model, and the selecting development files further comprises tracing references in SDK objects to determine the development files to select.

However, McInerney discloses in an analogous computer system identifying dependencies between development files (col. 3, lines 45-46 “Dependency analysis is automatic and is based on relations between components”); and wherein the identifying and creating generates the dependency model (col. 3, lines 29-30 “calculating the dependencies associated with a component”), and the selecting development files further comprises tracing references in SDK objects to determine the development files to select 4, lines 12-17 “The build mechanism assumes no preknowledge of dependencies... it "discovers" the dependencies of the components and keeps track of those dependencies... build mechanism will build a program from scratch when there is no preexisting dependency information”).

Art Unit: 2191

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of identifying dependencies between development files as taught by McInerney into the method of generating the modular code for operating system as taught by Keeley. The modification would be obvious because of one of ordinary skill in the art would be motivated to identify dependencies between development files to provide an optimize solution for mapping from source to object cod and vice versa as suggested by McInerney (col. 2, lines 34-56).

Claims 25 and 32 is the computer program product claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 26 is the computer program product claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 27 is the computer program product claim corresponding to method claim 4 and rejected under the same rational set forth in connection with the rejection of claim 4 above.

Claims 29, 30, and 31 are the computer program product claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 33 is the computer program product claim corresponding to method claim 12 and rejected under the same rational set forth in connection with the rejection of claim 12 above.

Art Unit: 2191

Claim 34 is the system claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 35 is the system claim corresponding to method claim 6 and rejected under the same rational set forth in connection with the rejection of claim 6 above.

Claim 36 is the system claim corresponding to method claim 9 and rejected under the same rational set forth in connection with the rejection of claim 9 above.

Claims 37 and 38 are the system claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 40 is the system claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 41 is the system claim corresponding to method claim 12 and rejected under the same rational set forth in connection with the rejection of claim 12 above.

10. Claims 5, 18, 22, 28, and 39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Keeley in view of US Patent No. 5,901,319 to Hirst (hereinafter called Hirst).

Per claims 5 and 22:

The rejection of claims 1 and 19 respectively, are incorporated, and further, neither Keeley nor McInerney explicitly disclose wherein the development files further comprise at least one or more of the following types of files: library files, documentation files, header files.

However, Hirst discloses in an analogous computer system the development files further comprise at least one or more of the following types of files: library files,

Art Unit: 2191

documentation files, header files (col. 8, lines 3-10 “The data structure 38, e.g., generic header file, is denoted by code block 120, and data structure 36 is denoted by code block 130... the instruction set includes an include command to the header file 120... number of function calls 114, 116, 118 and 119... function calls are defined by the selected generic defines contained within the second data structure 130”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method having one or more files types e.g. header, library, and documentation files as taught by Hirst into the method of generating the modular code for operating system as taught by Keeley. The modification would be obvious because of one of ordinary skill in the art would be motivated to include the header, library, and documentantation files in generating code to provide code more robust to run on different types of operating system as suggested by Hirst (col. 1, lines 28-63).

Claim 18 is the computer program product claim corresponding to method claim 5 and rejected under the same rational set forth in connection with the rejection of claim 5 above.

Claim 28 is the computer program product claim corresponding to method claim 5 and rejected under the same rational set forth in connection with the rejection of claim 5 above.

Claim 39 is the system claim corresponding to method claim 5 and rejected under the same rational set forth in connection with the rejection of claim 5 above.

Art Unit: 2191

Conclusion

11. Any inquiry concerning this communication or earlier communications from the examiner should be directed to **Satish S. Rampuria** whose telephone number is **(571) 272-3732**. The examiner can normally be reached on **8:30 am to 5:00 pm** Monday to Friday except every other Friday and federal holidays. Any inquiry of a general nature or relating to the status of this application should be directed to the **TC 2100 Group receptionist: 571-272-2100**

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, **Wei Y. Zhen** can be reached on **(571) 272-3708**. The fax phone number for the organization where this application or proceeding is assigned is **571-273-8300**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Satish S. Rampuria
Patent Examiner/Software Engineer
Art Unit 2191
12/27/2005


WEI Y. ZHEN
PRIMARY EXAMINER